Research Article

$k$ Nearest Neighbor Similarity Join Algorithm on High-Dimensional Data Using Novel Partitioning Strategy

Youzhong Ma,1,2 Qiaozhi Hua,3 Zheng Wen,4 Ruiling Zhang,1 Yongxin Zhang,1 and Haipeng Li5

1School of Information and Technology, Luoyang Normal University, Luoyang 471934, China
2Henan Key Laboratory for Big Data Processing & Analytics of Electronic Commerce, Luoyang 471934, China
3Computer School, Hubei University of Arts and Science, Xiangyang 441000, China
4School of Fundamental Science and Engineering, Waseda University, Tokyo 169-8050, Japan
5Capinfo Company Ltd., Beijing 100010, China

CorrespondenceshouldbeaddressedtoQiaozhiHua;11722@hbuas.edu.cn

Received 17 January 2022; Accepted 22 March 2022; Published 26 April 2022

Academic Editor: Thippa Reddy G

Copyright © 2022 Youzhong Ma et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

$k$ nearest neighbor similarity join on high-dimensional data has broad applications in many fields; several key challenges still exist for this task such as “curse of dimensionality” and large scale of the dataset. A new dimensionality reduction scheme is proposed by using random projection technique, then we design two novel partition strategies, including equal width partition strategy and distance split tree-based partition strategy, and finally, we propose $k$ nearest neighbor join algorithm on high-dimensional data based on the above partition strategies. We conduct comprehensive experiments to test the performance of the proposed approaches, and the experimental results show that the proposed methods have good effectiveness and performance.

1. Introduction

With the rapid development of emerging technologies such as big data [1, 2], Internet of things [3, 4], Deep Learning [5, 6], Adversarial Training [7, 8], Federated Learning [9], and 5G [10–12], the smart healthcare systems are becoming more and more pervasive and necessary in modern hospitals, and massive and diverse medical data have been accumulated by using a great volume of wearable sensors, the Internet of Medical Things [13, 14], or the Internet of Health Things [15, 16]. Medical data analysis, security, and privacy protection [17] are very important for using massive medical data. Many research works have been done, which can be used as the references to analyze the medical data and secure the Internet of Medical Things, such as the blockchain-based security approaches [18, 19], the security technologies in Internet of Internet of Vehicles [20, 21], Industrial Internet of Things [22–25], Artificial Intelligence of Things [26–28], Energy Internet [29], and Intelligent transportation [30, 31].

Similarity join operation plays an important role in medical data analysis [32, 33]. Threshold-based similarity join query on high-dimensional data can obtain all the data pairs whose distance is less than or equal to the given distance threshold, and it needs to know the distance threshold in advance; however, in many cases, it is hard or even impossible to get the distance threshold in advance, while $k$ nearest neighbor similarity join ($k$ NNI) does not need to obtain the distance threshold in advance. $k$ NNI is always used as the pre-processing stage for classification or clustering task and has broad real applications in many data mining tasks, such as multimedia analysis, spatial data mining, time series, data streams, and social network. Taking similar medical image pairs detection as an example, in some cases, it is hard to make a definite diagnosis according to the medical images only; however, it is possible for us to obtain some similar medical images of the existed confirmed cases ($k$ nearest neighbors) to help the doctors to make a final judgement on the unconfirmed diseases.

In many applications, the target objects can be represented as vector forms through feature extraction in order to
facilitate data processing, such as time series, videos, and trajectories, especially for the image processing tasks [34, 35]. With the continuous improvement of the accuracy of data acquisition equipment, the dimensionality of vectors representing the target objects will be very high, maybe hundreds of dimensions or even tens of thousands of dimensions. The calculation of the similarity or distance between the object pairs is a time costly operation. Most of the existed approaches conduct k NN join operation directly on the original dimensional space, so their performance is not ideal. It is an effective way to reduce the time cost of k NN join through reducing the dimensionality of the original data points. There are three contributions in the paper:

(i) We proposed an effective dimensionality reduction approach that can make sure that the data points in the projected space preserve the location relationship to some extent as in the original space.

(ii) We proposed two partition strategies, including equal width partition strategy and distance split tree-based partition strategy, and a novel k nearest neighbor join algorithm was proposed by using the above two partition strategies.

(iii) Comprehensive experiments are conducted, and the final results prove that our approaches have better effectiveness and performance.

The other parts of the paper are arranged as follows. The detailed related research works are described in Section 2. Section 3 displays the notations, problem definitions, and some theorems. The lower bound probability is figured out in Section 4. The k nearest neighbor join algorithm using random projection and partition strategies is described in Section 5. The detailed experimental results of the proposed approaches are displayed in Section 6. Section 7 makes a conclusion of the paper and points out the future research directions.

2. Related Works

Many researchers have conducted in-depth research on similarity join problems because of their broad applications and important role in data mining or machine learning context; several survey articles have conducted a comprehensive and detailed analysis of the similarity join problem literature [36, 37].

k nearest neighbor join: an approximated k NN similarity join approach in metric spaces was proposed by Ferrada et al. [38], its time complexity is \( \Theta(n^{3/2}) \), and the empirical precision can reach up to 46%. Lu et al. [39] designed a novel approach called PCBJ by using Voronoi diagram, which can deal with exact k NN similarity join problems; however, its performance is not very good as the increase of data dimensionality. Dai et al. [40] proposed two novel k NN join algorithms based on the MapReduce framework, which are DSGMP-J using Distributed Sketched Grid and VDMP-J using Voronoi diagram; DSGMP-J [40] approach is easy to implement, but it ignores the real distribution of the dataset. Zhao et al. [41] designed an effective data partitioning scheme called k NN-DP, which can solve the load imbalance issues caused by the data skewness problem; two novel schemes called LSH+ and z-value+ were developed based on k NN-DP to deal with k NN join operations under MapReduce framework. Song et al. [42] conducted a detailed analysis of the common workflows of the several existing k NN algorithms and further analyzed their load balancing, accuracy rate, and overall complexity; finally, a choice guideline was given which can help select the suitable methods for a specific case. Song et al. [43] also conducted a detailed comparison among the existing k NN join approaches both theoretically and experimentally on the MapReduce platform. RankReduce approach [44] was proposed by using locality-sensitive hashing with MapReduce for processing k nearest neighbor query. Hu et al. [45] proposed an adaptive v k NN join algorithm by using the Voronoi diagram, which can eliminate many unnecessary computations. There still exist many other research works about k NN join problems in several applications, such as Trajectory Data [46], machine learning [47], and Hilbert R-tree-based k NN join algorithms [48].

Top-k similarity join: Kim et al. [49] proposed two serial algorithms called the “divide-and-conquer algorithm” and “branch-and-bound algorithm” using the MapReduce framework, which can deal with Top-k similarity join problems efficiently. Chen et al. [50] developed a new distance function based on LSH and proposed an R-tree-based Top-k similarity join algorithm using the Spark framework, and the test results proved that the RDD-based algorithm has better scalability and effectiveness than that of Hadoop. Ma et al. [51] developed a Top-k threshold estimation approach through sampling and designed an effective filtering solution by using the Symbolic Aggregate Approximation technique and then proposed a SAX-Top-k algorithm, which can deal with Top-k similarity join problem. Lei et al. [52] explored the similarity join problems for massive probabilistic dataset. The main idea of Lei et al. [52] is mapping the probabilistic data from the original space to the reduced dimensional space (one dimension), and then the range query on the one-dimensional space can be instead of the threshold-based similarity join query on the original space. Based on the above schemes, the authors proposed the Top-k Block Nested Loop Join Algorithm and Top-k Data Locality Preserving Join Algorithm, respectively. MELODY-JOIN [53] can improve the efficiency of the Top-k join on the histogram probabilistic dataset by using the standard lower bound space of the EMD distance; however, it cannot deal with the data skew problem efficiently. EMD-MPJ [54] proposed a novel Mapping-based Data Partitioning Framework that can solve the data skew problem. Heads-Join [55] made an extension to MELODY-JOIN [53] so that it can deal with both range similarity join and Top-k similarity efficiently.

Threshold-based similarity join: Cristiani et al. [56] designed a novel randomized set join method whose recall can be up to 100%, and its performance is better than that of the existing approach theoretically and empirically. Gowanlock et al. [57] proposed several novel methods to accelerate the similarity self-join by making full use of the
power and characteristic of GPU. There still exist some other research works which focus on the similarity join problem using GPU [58]. Sandes et al. [59] developed a novel filtering scheme that can speed up the exact set similarity join more efficiently. Ding et al. [60] exploited the privacy preserving problems in similarity join using MapReduce context. Wu et al. [61] proposed a novel parallel framework called SMS-Join which can support similarity join operations in metric space using the MapReduce paradigm. Ma et al. studied the similarity join problems for high-dimensional dataset through developing a novel dimension reduction approach based on the Piecewise Aggregate Approximation technique and proposed two algorithms called SAX-Based HDSJ [62] and Mpv-V-SJQ [63].

3. Preliminaries

3.1. Notations. The notations used in this paper are listed in Table 1.

3.2. Problem Definition. The definitions of KNN and KNN join will be described in this subsection. Given two datasets \( R \in \mathbb{R}^d \) and \( S \in \mathbb{R}^d \), \( R = \{ r_1, r_2, \ldots, r_n \} \), \( S = \{ s_1, s_2, \ldots, s_n \} \), \( |R| = n_1 \), and \( |S| = n_2 \). \( r_i \) is the \( i \)-th data point from \( R \), \( s_j \) is the \( j \)-th data point from \( S \), \( s_j = \{ s_{j1}, s_{j2}, \ldots, s_{jd} \} \), and the distance measurement used in the paper is the Euclidean distance denoted as \( \text{dist}(r_i, s_j) \),

\[
\text{dist}(r_i, s_j) = \sqrt{\sum_{l=1}^{d} (r_{il} - s_{jl})^2},
\]

where \( \text{dist}(r_i, s_j) > = 0 \) and \( \text{dist}(r_i, s_j) \) equals 0 when \( r = s \).

Definition 1. K Nearest Neighbor Join (KNN). For a \( d \)-dimensional dataset \( S \in \mathbb{R}^d \) and a query data point \( r \), the KNN operation of \( r \) on \( S \) can be recorded as \( \text{knn}(r, S, k) \) aiming to obtain the \( k \) nearest neighbors of \( r \) in \( S \):

\[
\text{knn}(r, S, k) = \{ s_1, s_2, \ldots, s_k | s_i \in S, 1 \leq i \leq k \}; \quad \text{for each } \forall s_j \in S - \{ s_1, s_2, \ldots, s_k \}, \text{the distance meets the following requirements:}
\]

\[
\text{dist}(r, s_1) \leq \text{dist}(r, s_2) \leq \cdots \leq \text{dist}(r, s_k) \leq \text{dist}(r, s_j).
\]

Definition 2. K Nearest Neighbor Similarity Join (KNN Join). Given two datasets \( R \in \mathbb{R}^d \) and \( S \in \mathbb{R}^d \), the KNN similarity join operation on \( R \) and \( S \) can return the \( k \) nearest neighbors for each data point \( r \in R \) from \( S \), which can be denoted as \( \text{knn}(\langle r, \text{knn}(r, S, k) \rangle) \) for each \( r \in R \).

3.3. Theorems

Theorem 1. Given two \( d \)-dimensional data points \( s_1 \) and \( s_2 \), then

\[
\text{dist}(s_1, s_2) = g(s_1) - g(s_2)/\text{dist}(s_1, s_2) \sim N(0, 1).
\]

Theorem 2. Given two \( d \)-dimensional data points \( s_1 \) and \( s_2 \),

\[
\text{dist}(s_1, s_2) \sim \chi^2(m).
\]

Theorem 3. If \( \text{dist}(s_1, s_2) \leq k \), then the probability that \( \Delta_m^{(s_1, s_2)} \) is less than or equal to \( k \) will be bigger than or equal to \( 1 - P(\chi^2 > k^2) \), which can be denoted as follows:

\[
P(\Delta_m^{(s_1, s_2)} \leq k | \text{dist}(s_1, s_2) \leq k) \geq 1 - P(\chi^2 > k^2).
\]

Theorem 4. Given three \( d \)-dimensional data points \( r, s_1, s_2 \),

\[
\text{F} = U/V \sim F(m, m).
\]

Proof. According to Theorem 2, \( U \sim \chi^2(m) \) and \( V \sim \chi^2(m) \); that is to say, \( U \) and \( V \) both obey the \( \chi^2 \) distribution with freedom \( m \).

Based on the definition of \( F \) distribution, \( \text{F} = U/V \) obeys the \( F \) distribution with degrees of freedom \((m, m)\); that is, \( \text{F} \sim F(m, m) \).

Theorem 5. If \( \text{dist}(r, s_1) \leq \text{dist}(r, s_2) \), then the probability that \( \Delta_m^{(r, s_1)} \leq k \Delta_m^{(r, s_2)} \) is bigger than \( 1 - P(F > k^2) \); that is, \( P(\Delta_m^{(r, s_1)} \leq k \Delta_m^{(r, s_2)} | \text{dist}(r, s_1) \leq \text{dist}(r, s_2)) > 1 - P(F > k^2) \). \( F \) is the distribution with degrees of freedom \((m, m)\); that is, \( \text{F} \sim F(m, m) \).

Proof.
\begin{table}
\centering
\caption{Notations.} 
\begin{tabular}{l|l}
Notation & Meaning of the notation \\
\hline
\(n_1,n_2\) & The data points’ number in the dataset. \\
e & The width of each partition under equal width partition strategy. \\
d & The data point’s dimensionality. \\
\(\text{dist}(s_1,s_2)\) & The Euclidean distance of data point \(s_1\) and data point \(s_2\). \\
g(s) & \(g(s) = a \cdot s\), \(a\) is a \(d\)-dimensional vector, and each element is a random variable that obeys \(\text{p-stable distribution.}\) \\
\(\pi_m(s)\) & \(\pi_m(s) = \langle g_1(s), g_2(s), \ldots, g_m(s) \rangle\). \\
\(\Delta_m(s_1,s_2)\) & \(\Delta_m(s_1,s_2) = \text{dist}(\pi_m(s_1),\pi_m(s_2))\). \\
\(\chi^2(m)\) & Chi-square distribution with degree of freedom \(m\). \\
\(\pi_1(s)_{\text{min}}\) & The minimum projected value of data point \(s\). \\
\(\pi_1(s)_{\text{max}}\) & The maximum projected value of data point \(s\). \\
\(\text{len}\) & The width of all the projected values in one-dimensional space; that is, \(\pi_1(s)_{\text{min}} = \pi_1(s)_{\text{max}}\). \\
\(PN\) & The number of the partitions in one-dimensional space. \\
\(P_i\) & The \(i_{\text{th}}\) partition. \\
\end{tabular}
\end{table}

\[
\Delta_m(r,s) \geq 0 \text{ and } \Delta_m(r,s) \geq 0,
\]
\[
\therefore P(\Delta_m(r,s) \leq k \Delta_m(r,s)) \leq \text{dist}(r,s) = \text{dist}(r,s)
\]
\[
= P\left(\frac{\Delta_m^2(r,s)}{\text{dist}^2(r,s)} \leq k^2 \frac{\Delta_m^2(r,s)}{\text{dist}^2(r,s)} \right)
\]
\[
= P\left(\frac{\Delta_m^2(r,s)}{\text{dist}^2(r,s)} \leq k^2 \frac{\Delta_m^2(r,s)}{\text{dist}^2(r,s)} \right)
\]
\[
= 1 - P\left(\frac{\Delta_m^2(r,s)}{\text{dist}^2(r,s)} > k^2 \frac{\Delta_m^2(r,s)}{\text{dist}^2(r,s)} \right).
\]
\[
\therefore \frac{\Delta_m^2(r,s)}{\text{dist}^2(r,s)} \sim \chi^2(m) \text{ and dist}(r,s) \leq \text{dist}(r,s).
\]
\[
P(\Delta_m(r,s) \leq k \Delta_m(r,s)) \leq \text{dist}(r,s) \leq \text{dist}(r,s)
\]
\[
> 1 - P\left(\frac{\Delta_m^2(r,s)}{\text{dist}^2(r,s)} \leq k^2 \frac{\Delta_m^2(r,s)}{\text{dist}^2(r,s)} \right)
\]
\[
= 1 - P\left(\frac{\Delta_m^2(r,s)}{\text{dist}^2(r,s)} > k^2 \right).
\]
\[
\text{ According to theorem 4, } \frac{\Delta_m^2(r,s)}{\text{dist}^2(r,s)} \sim F(m,m),
\]
\[
P(\Delta_m(r,s) \leq k \Delta_m(r,s)) \leq \text{dist}(r,s) \leq \text{dist}(r,s)
\]
\[
> 1 - P\left(\frac{\Delta_m^2(r,s)}{\text{dist}^2(r,s)} \leq k^2 \right).
\]
\[
\text{ We can conclude that if } s_1 \text{ is closer to } r \text{ than } s_2 \text{ in the original } d\text{-dimensional space, } \pi_m(s_1) \text{ is still likely to be closer to } \pi_m(r) \text{ than } \pi_m(s_2) \text{ in the projected } m\text{-dimensional space with lower bound probability } 1 - P(F(m,m) > 1). \qedhere
\]
4. Probability Computation

When \( m = 1, k = 1 \), the probability \( P(F(m, m) > 1) \) can be figured out based on the probability density of \( F \) distribution which is described as follows:

\[
\psi(y) = \begin{cases} 
\frac{1}{\Gamma(n_1/2)\Gamma(n_2/2)} \left[ 1 + (n_1/n_2) \right]^{(n_1+n_2)/2} y^{(n_1+n_2)/2-1}, & y > 0, \\
0, & \text{other.}
\end{cases}
\]

(3)

Given the freedom \((1,1)\) and the upper quantile with 1, the probability \( P(F(m, m) > 1) \) can be calculated as the follows:

\[
P(F > 1) = \int_1^\infty \psi(y)dy = \int_1^\infty \frac{1}{\sqrt\pi} \frac{y^{-1/2}}{(1+y)^{1/2}} \frac{1}{\Gamma(1/2)} \frac{1}{\Gamma(1/2)} d\psi
\]

(4)

\[
P(F > 1) = \frac{2}{\pi} \frac{\pi - \pi}{4} = 0.5.
\]

The result of the above computation shows that, given three \( d \)-dimensional data points \( r, s_1, s_2 \), and \( s_1, r, s_1, s_2 \in \mathbb{R}^d \), they can be reduced to 1-dimensional space through dot product with a \( d \)-dimensional vector \( a \). If \( s_1 \) is closer to \( r \) than \( s_2 \) in the original \( d \)-dimensional space, the lower bound probability that \( \pi_1(s_1) \) is still likely to be closer to \( \pi_1(r) \) than \( \pi_1(s_2) \) in the projected 1-dimensional space is 0.5.

5. \( k \) Nearest Neighbor Join Using Novel Partitioning Strategy

5.1. Algorithm for \( k \) Nearest Neighbor Join Using Novel Partitioning Strategy. Theorem 5 shows that if \( s_1 \) is closer to \( r \) than \( s_2 \) in the original \( d \)-dimensional space, \( \pi_m(s_1) \) is still likely to be closer to \( \pi_m(r) \) than \( \pi_m(s_2) \) in the projected \( m \)-dimensional space with the lower bound probability \( 1 - P(F(m, m) > 1) \). The conclusion implies that data points in projected space maintain relative location relationship as in original dimensional space. So we proposed \( k \) nearest neighbor join algorithm using random projection (RP \( k \) NN); it includes two main stages: the first stage is responsible for dimension reduction and space partition, and the second stage is used to conduct \( k \) NN join in reduced dimensional space. Figure 1 shows the general framework of \( k \) nearest neighbor similarity join algorithm using novel partitioning strategy. The concrete process of the proposed algorithm is described in Algorithm 1. The getPartition routine projects all the data points into one-dimensional space and divides the data points into several partitions according to the specific partition strategy (line 1). For each partition \( P_i \), its corresponding partition \( P_i^\pi \), which needs to be compared with \( P_i \), can be obtained through lines 3–9. If \( P_i \) is the leftmost partition, \( P_i^\pi \sim P_i \cup P_{i+1} \) (line 5). If \( P_i \) is the rightmost partition, \( P_i^\pi \sim P_{i-1} \cup P_i \cup P_{i+1} \) (line 7); in other cases, \( P_i^\pi \sim P_{i-1} \cup P_i \cup P_{i+1} \) (line 9). Finally, for each data point \( v \in P_i \), \( k \) NN join routine is used to find its \( k \) nearest neighbors from \( P_i \) (lines 10–12).

5.2. Partition Strategy

5.2.1. Equal Width Partition Strategy. All the data points are divided into several partitions with equal width. The total partition number can be set to \( PN = \lfloor \sqrt{n} \rfloor \). Suppose that \( \pi_1(s)^{\text{min}} \) is the minimum projected value of \( s \) in one-dimensional space and \( \pi_1(s)^{\text{max}} \) is the maximum projected value of \( s \) in one-dimensional space; that is \( \pi_1(s)^{\text{min}} = \min \{ \pi_1(s), s_j \in \mathbb{R}^d \} \) and \( \pi_1(s)^{\text{max}} = \max \{ \pi_1(s), s_j \in \mathbb{R}^d \} \); \( \text{len} \) is the width of all the projected values in one-dimensional space, \( \text{len} = \pi_1(s)^{\text{max}} - \pi_1(s)^{\text{min}} \); \( \epsilon \) is the width of each partition, \( \epsilon = \lfloor \text{len}/PN \rfloor \); given a data point \( s \), its corresponding partition number is \( P_i = \lfloor \pi_1(s)/\epsilon \rfloor \). The detailed procedure can be shown in Figure 2 and Algorithm 2.

5.2.2. Distance Split Tree-Based Partition Strategy. The previous equal width partition strategy is easy to implement; however, it cannot deal with skewed dataset efficiently. According to our proposed approach, the \( d \)-dimensional data point \( s \) will be projected into one-dimensional space through \( \pi_1(s) = g_1(s) = \cdot \cdot \cdot a_i \cdot a_i \cdot a_i \cdot a_i \cdot a_i \) each element of \( a \) obeys standard normal distribution; that is: \( a_i \sim N(0, 1), i \in [1, d] \). The projected value \( \pi_1(s) \) is subject to normal distribution, so it is skewed. Figure 3 shows the distribution of the projected values.

Theorem 6. Given two \( d \)-dimensional data points \( s, a \), \( a_i \sim N(0, 1), i \in [1, d] \), \( \pi_1(s) = g_1(s) = s \cdot a = \sum_{i=1}^{d} s \cdot a_i \), and then \( \pi_1(s) \sim N(0, \sum_{i=1}^{d} a_i^2) \).
Input: R, k //dataset, the number of nearest neighbors to find
Output: res //a set of pairs of data points
(1) partitions—get Partition(R, n);
(2) res=Ø;
(3) for i = 1; i ≤ [partitions]; i + + do
(4) if i = 1 then
(5) $P_i = P_{i-1} \cup P_i \cup P_{i+1}$
(6) else if i = [partitions] then
(7) $P_i = P_{i-1} \cup P_i$
(8) else
(9) $P_i = P_{i-1} \cup P_i \cup P_{i+1}$
(10) for data point v ∈ $P_i$ do
(11) temp=kNN Join(k, v, $P_i$)
(12) res=res∪temp
(13) return res.

**Algorithm 1:** Algorithm for k nearest neighbor join using novel partitioning strategy.
Input: $R$, $n$ //dataset, the cardinality of the dataset
Output: partitions //list of the sets that partition the dataset

1. $PN = \lceil \sqrt{n} \rceil$
2. $\text{partitions}[i] \leftarrow \emptyset, \forall i, 0 \leq i \leq PN$
3. $\pi_1(s)_{\text{min}} = \min \{ \pi_1(s), s_j \in R \}$
4. $\pi_1(s)_{\text{max}} = \max \{ \pi_1(s), s_j \in R \}$
5. $\text{len} = \pi_1(s)_{\text{min}} - \pi_1(s)_{\text{max}}$
6. $\epsilon = \lceil \text{len}/PN \rceil$
7. choose one vector randomly, recorded as $a, \forall v \in a \sim N(0,1)$
8. //divide $R$ into $PN$ partitions according to $P_i$, the data points which have the same $P_i$,
9. //belong to the same partition, recorded as partitions[1], partitions[2], ..., partitions[PN].
10. for each datapoint $s \in R$ do
11. $\pi_1(s) \leftarrow \langle g_1(s) \rangle$
12. $P_i = \lceil \pi_1(s)/\epsilon \rceil$
13. partitions[$P_i$] $\leftarrow \langle P_i, \pi_1(s), s \rangle$

Algorithm 2: Equal width partition strategy.

Figure 3: The distribution of the projected values ($\pi_1(s)$): (a) $\text{dim} = 128$ and data science $= 30000$; (b) $\text{dim} = 256$ and data science $= 30000$; (c) $\text{dim} = 512$ and data science $= 30000$; (d) $\text{dim} = 1024$ and data science $= 30000$. 
Proof.

\[ a_i \sim N(0, 1), \]
\[ s_i a_i \sim N(0, s_i^2), \]
\[ \pi_1(s) = \sum_{i=1}^{d} s_i a_i, \]
\[ \Rightarrow \pi_1(s) \sim N\left(0, \sum_{i=1}^{d} s_i^2\right), \]
\[ \Rightarrow \pi_1(s) \text{ is subject to normal distribution and its variance is } \sum_{i=1}^{d} s_i^2. \]

Aiming to deal with the data skew problem, we proposed a novel partition strategy called distance split tree (DST-) based partition strategy. Figure 4 displays the structure of DST. The main idea of DST is that after the original d-dimensional data points are mapped into one-dimensional space, in the beginning, all the data points are divided into equal width partitions with threshold \( e = \pi_1(s)_{\text{max}} - \pi_1(s)_{\text{min}} / 2 \), \( e \) is an adjustable parameter. maxNum is the upper bound of data point count contained in each partition. Once the data point count in a specific partition exceeds maxNum, the partition will be divided into two new partitions with equal width again and so on, and finally, a distance split tree is formed. For each leaf node, the level of the node’s hierarchy, the node number in the specific level, the count of the data, and the corresponding dataset are recorded. Based on the above information, the distance range corresponding to each leaf node can be calculated. The corresponding distance width of each node in the current level can be calculated: \( 1/2^{\text{level} - 1} \epsilon \). The corresponding distance range of the node can be calculated by or de rNo in the level (order No - 1/2^{level - 1}ε, order No/2^{level - 1}ε). Thus, the distance range corresponding to the N2 node can be calculated: \( [3 - 1/2^{\text{level} - 1}ε, 3/2^{\text{level} - 1}ε] \); that is, \( [2/4ε, 3/4ε] \).

The construction of distance split tree: the construction process of the distance split tree is as follows: firstly, build a root node \( N_{\text{root}} \), for each data point \( s_i \in R \), and figure out its projected value in one-dimensional space \( \pi_1(s) \). Then all the data points are divided into equal width partitions with threshold \( \epsilon \), and the corresponding partition number of each vector \( s_i \) in the mapping space is obtained \( pi \ d \sim [\pi_1(s_i)/\epsilon] \). If the node with the number \( pi \ d \) does not exist, a new child node with the number \( pi \ d \) will be generated. If it already exists, \( s_i \) is inserted into the node \( pi \ d \), and its count value is increased by 1. Once the amount of data point in a node exceeds a given threshold, such as \( \text{maxNum} \), the node will be further divided into two subnodes according to the distance range. Repeat this procedure, and finally, a distance split tree is generated.

Data partitions generated: after the distance split tree is constructed, the partitions set can be obtained through preorder traversal for distance split tree; only the leaf nodes are left as the member of the final partitions set. Then the obtained partitions set can be used in Algorithm 1. \( \square \)

6. Time Complexity Analysis

In this section, we mainly analyze the time complexity of our proposed method. Given the \( d \)-dimensional dataset \( R \) and \( |R| = n \), the total partition number is \( PN = \lfloor \sqrt{n} \rfloor \), \( P_i \) represents the \( i_{th} \) partition, and Cost represents the total computations of the proposed method. The time complexity is as the follows:

\[ \text{Cost} = \sum_{i=1}^{PN} |P_i| \cdot |P_i \cup P_{i-1} \cup P_{i+1}| \cdot d. \]

In the best cases, all the data points in \( R \) are evenly distributed in each partition; that is, \( |P_i| = \lfloor \sqrt{n} \rfloor \), so

\[ \text{Cost} = \sum_{i=1}^{PN} \lfloor \sqrt{n} \rfloor \cdot 3 \cdot \lfloor \sqrt{n} \rfloor \cdot d \]
\[ = 3 \cdot \sqrt{n} \cdot \lfloor \sqrt{n} \rfloor \cdot \lfloor \sqrt{n} \rfloor \cdot d \]
\[ = 3 \cdot n^{3/2} \cdot d. \]

The time complexity in the best case can be recorded as \( \Theta(n^{3/2}d) \).

In the worst case, supposing that all the data points are included in one partition, then the time complexity should be \( \Theta(n^2d) \). Because the projected values of the proposed method obey normal distribution, it is between the best case and the worst case, so the time complexity of the proposed method lies in \( (\Theta(n^{3/2}d), \Theta(n^2d)) \).

7. Experimental Analysis

We conducted experiments to test the effectiveness and performance of the proposed methods, \( k \) nearest neighbor join algorithm using random projection with equal width partition strategy (RP k NNEW) and \( k \) nearest neighbor join algorithm using random projection with distance split tree-based partition strategy (RP k NNNDST), and made comparisons between our proposed methods and the existing
methods including a KNN [38] and the brute force method with Block Nested Loop Join Strategy (BNLJ). a KNN [38] is a relatively new research work on k NN similarity join problem, and it also adopted the algorithms based on the partitioning strategy.

Experimental settings: our tests are performed on HP workstation, and the configurations are as follows: CPU: Intel Xeon Gold 6136 @ 3.00 GHz, memory: 128 GB, disk: 2 TB, OS: 64-bit Windows 10, and 12 cores. Table 2 describes the parameters and their values. The bold fonts represent the default values.

Datasets: the datasets adopted in our experiments are synthetic data, the elements of the vector are uniformly distributed in the range [0, 1], and the dimensionality of the datasets includes 128, 256, 512, and 1024. The datasets are listed in Table 3.

7.1. Precision versus Data Size. It can be concluded that RP k NNEW has the best precision among the above three approaches, including a KNN, RP k NNEW, and RP k NNDST; in some cases, the precision of RP k NNEW is more than 50%. However, Figure 5 shows that the precision of a KNN, RP k NNEW, and RP k NNDST is not very stable for different data size, and the precision of RP k NNDST is between that of a KNN and RP k NNEW.

7.2. Precision versus Dimension. Figure 6 displays the precision of a KNN, RP k NNEW, and RP k NNDST under different dimension. The proposed method RP k NNEW has the best precision; although the precision of RP k NNEW varies under different dimension, it is always higher than 40% under all conditions. The precision of RP k NNDST is relatively stable under different dimension, and it is lower than that of RP k NNEW. Because RP k NNDST adopts the distance tree-based partition strategy and all the data points will be distributed into different partitions more evenly, every partition will not contain much more data points, so its precision will decrease to some extent compared with RP k NNEW. The precision of our proposed methods, including RP k NNEW and RP k NNDST, is better than that of a KNN under all different dimension.

7.3. Precision versus k. Figure 7 displays the precision of the above approaches (including a KNN, RP k NNEW, and RP k NNDST) under different k. The results prove that the precision of RP k NNEW and RP k NNDST is higher than that of the existing method a KNN. The precision of RP k NNEW is higher than that of RP k NNDST; the reason is that RP k NNEW adopted the equal width partition strategy; while the projected values are skewed, several partitions will contain more data points; the precision will be higher accordingly. The precision of a KNN and RP k NNDST is basically stable under different k value, while the precision of RP k NNEW is a little more sensitive to the different k value.

7.4. Precision Distribution. Figure 8 displays the precision distribution of a KNN, RP k NNEW, and RP k NNDST. It can be found that the precision of some data points is very low (less than 5%), and the precision of some data points is very high (more than 80%) by using RP k NNEW. The average precision of RP k NNEW is higher than that of a KNN and RP k NNDST; the percentage of the data points whose precision is more than 80% is 20.6%, 0.05%, and 0.57%, respectively, for RP k NNEW, a KNN, and RP k NNDST. The main reason is that RP k NNEW adopts the equal width partition strategy, which cannot deal with the skewed projected values effectively; however, RP k NNDST adopts distance split tree-based partition strategy, which can distribute all the data points into different partitions more evenly.

7.5. Performance versus Data Size. Figure 9 displays the performance of BNLJ, a KNN, RP k NNEW, and RP k NNDST on the datasets with different size. The run time of a KNN, RP k NNEW, and RP k NNDST is much less than that of the BNLJ method; the run time of BNLJ increases exponentially with the size of the datasets; however, the run time of a KNN, RP k NNEW, and RP k NNDST increases.
linearly with the size of the datasets. The run time of RP $k$ NNEW is a little bit more than that of $a$ KNN and RP $k$ NNDST. While we can find that RP $k$ NNEW has the best precision among all the methods based on the above precision analysis, we can choose the RP $k$ NNEW method when the performance requirements are not very strict; otherwise, RP $k$ NNDST will be the most appropriate choice, because the precision of RP $k$ NNDST is higher than that of $a$ KNN, while its run time is less than that of RP $k$ NNEW.

### 7.6. Performance versus Dimension

Figure 10 displays the performance of our proposed methods and the existing methods for different dimensions, which are 128, 256, 512, and 1024, respectively. The time of all algorithms grows with the increase of the dimension, and the reason is the bigger the dimension, the higher the time complexity. The performance of RP $k$ NNEW is the best when the dimension is less than 512, while the run time of RP $k$ NNEW will be slightly higher than that of $a$ KNN when the dimension exceeds 512. The run time of RP $k$ NNEW is higher than that of $a$ KNN and RP $k$ NNEW, while Figure 6 shows that RP $k$ NNEW has the best precision in all cases.

### 7.7. Performance versus $k$

The performance of BNLJ, $a$ KNN, RP $k$ NNEW, and RP $k$ NNDST with different $k$ (data size = 30000; dim = 512) is displayed in Figure 11. The run
Figure 6: Precision versus dimension.

Figure 7: Precision versus $k$.

Figure 8: Continued.
Figure 8: The distribution of the precision: (a) dim = 512, data size = 30000, and k = 30; (b) dim = 512, data size = 30000, and k = 30; (c) dim = 512, data size = 30000, and k = 30; (d) dim = 1024, data size = 30000, and k = 30.

Figure 9: Performance versus data size.

Figure 10: Performance versus dimension.
time of all the above algorithms changes little with the different value of $k$. The run time of RP $k$ NNDST is very close to that of a KNN, and the reason has been explained in Section 6; however, the precision of RP $k$ NNDST is better than that of a KNN according to Figure 7. The run time of RP $k$ NNDST and a KNN is less than that of RP $k$ NNEW.

7.8. Preprocessing Time. Figure 12 displays the preprocessing time required by the construction of the distance split tree (DST) in RP $k$ NNDST approach for different dataset. The preprocessing time increases almost linearly with the size of the dataset. Given a dataset with a fixed size, the preprocessing time increases exponentially with the growth of the dimension. Overall, the proportion of preprocessing time in the total time is low and relative stable. The benefit of the distance split tree (DST) can make up for the additional overhead caused by the construction of DST and can make the total run time of RP $k$ NNDST less than that of RP $k$ NNEW.

8. Conclusions

In the above sections, we mainly studied the $k$ nearest neighbor similarity join problem on high-dimensional data. We proposed $k$ nearest neighbor join algorithm using random projection with equal width partition strategy (RP $k$ NNEW) and $k$ nearest neighbor join algorithm using random projection with distance split tree-based partition strategy (RP $k$ NNDST), which can filter out many unnecessary comparisons and ensure the required precision. We also conducted several experiments to test the effectiveness and performance of our proposed approaches, and the test results show that the proposed approaches in this paper have better effectiveness and performance. However, the proposed approaches in this paper have some limitations, and they can only work with the Euclidean distance. In future research works, we are planning to further study the $k$ NN similarity join approaches, which can deal with other similarity measures, other more effective dimension...
Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was partially supported by the grants from the Science and Technology Research Plan Project of Henan Province (202102210357); the Innovative Research Team (in Science and Technology) at University of Henan Province (202102210357); the Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research (KAKENHI) (JP21K17737); and the Hubei Natural Science Foundation (2021CFB156).

References


